Modeling the Effects on Power and Performance from Memory Interference of Co-located Applications in Multicore Systems

Daniel Dauwe[1], **Ryan Friese**[1],
Sudeep Pasricha[1,2], **Anthony A. Maciejewski**[1], **Gregory A. Koenig**[3], **and Howard Jay Siegel**[1,2]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science
Colorado State University
Fort Collins, CO, 80523
[3]Oak Ridge National Laboratory
Oak Ridge, TN 37830

Abstract— *In this study, we analyze interference trends when co-running multiple applications possessing varying degrees of memory intensity on multi-core processors. We conduct tests with PARSEC benchmark applications and explore energy consumption, execution times, and main memory accesses when interfering applications share last-level cache. We also explore how co-running applications are impacted when the processor frequency is modified using dynamic voltage and frequency scaling (DVFS). A portable and lightweight testing framework is presented and results are shown for experiments conducted on an Intel i7 quad-core system. It is shown that the degree of degradation due to co-location interference on execution time is highly dependent on the types and number of applications co-located on cores that share the last-level cache.*

Keywords: memory interference; application co-location; benchmarking; energy-aware computing; dynamic voltage and frequency scaling; multi-core processors

1. Introduction

There is an ever present desire to increase the performance and capabilities of current high performance computing systems. Frequently, increased performance comes at the cost of increased power dissipation, which has become a major challenge in large scale computing systems. According to the 2012 DatacenterDynamics census [DcD12], global datacenter power requirements in 2007 were 12 GW, but doubled to 24 GW in 2011. Then, in 2012 the power requirements grew by 63% to 38 GW [DcD12]. There is thus a critical need to profile and characterize power dissipation in computing nodes, as a precursor to developing techniques that can minimize power dissipation.

The use of multiple cores in today's processing units is commonplace as parallel processing remains a popular technique for speeding up the execution time of workloads. In an ideal system, doubling the number of cores in a server doubles the performance as long as the workload is perfectly parallel. However, cores in today's processors typically share resources in some manner (such as last-level cache, DRAM, network, and storage), causing contention for these resources and degrading performance, potentially resulting in larger amounts of energy being consumed.

Understanding the effects of resource contention caused by co-locating applications in multi-core processors is becoming increasingly important as the number of cores per processor continues to increase. Co-location occurs when more than one application is executing and sharing resources on a multi-core processor. Different applications have different resource requirements, and by studying the execution time and energy effects across several applications we can better understand the implications associated with co-locating applications on multi-core processors. A better understanding of these effects becomes critical to intelligently mitigating interference and improving performance and energy consumption.

This work examines the effects of memory interference on energy usage and application execution time (or application throughput). Memory interference is introduced by executing a number of applications across the cores of a given processor, resulting in the applications sharing L2 or L3 caches in the memory hierarchy. Specifically, the research provides a testing infrastructure that monitors and collects data on the following attributes as various applications are executed on a multi-core processor: energy usage, execution time, and main memory (DRAM) accesses. Applications that access main memory frequently can be considered to be "memory intensive." This research also analyzes how applications with varying levels of memory intensity affect the attributes of applications with which they are co-located.

We consider the attributes of four different workloads taken from the PARSEC benchmarking suite [Par14] for measurement and analysis. The four workloads were selected to represent a range of applications with varying memory intensity. The workloads are executed on a quad-core Intel i7 processor. We chose to conduct experiments on this processor because it is being currently used in compute nodes within several datacenters that are increasingly deploying systems with low-end commodity processors (instead of high end server processors) to keep hardware costs manageable at a large scale. Results are first gathered for the case where each application executes by itself on a single core in the processor. These results are then used as a baseline to contrast the degradation in performance when there are two and four applications co-located on the cores of the processor. It is shown that for more memory intensive applications, there is a greater decrease in performance as memory interference increases. Additionally, this research incorporates the dynamic voltage and frequency scaling (DVFS) property of the processor, and runs all the tests across a range of different voltages and frequencies. The results from this research can be used to provide highly accurate execution time and energy consumption information for use in the area of resource allocation in high performance computing systems, where application tasks are co-located on the cores of multi-core processors[FrB13], [FrK13], [OxP13].

In summary, this work makes the following contributions: (a) proposes a portable benchmark testing environment capable of measuring and analyzing the effects of cross-application interference on energy consumption, execution time, and memory accesses of various applications across CPU frequencies; (b) provides an analysis for a set of real-world workload execution scenarios using this testing environment to perform interference tests on a modern quad-core machine; and (c) gives insights into how large computer systems based on multi-core processors may be able to improve their energy use based on the memory interference caused by co-located applications.

The remainder of this paper is organized as follows. The next section will discuss related work. Section 3 will describe the portable testing environment. The workloads being tested and the experimental setup will be explained in Section 4. Section 5 will present and provide analyses for the results of the experiments. Conclusions and plans for future work will be discussed in Section 6.

2. Related Work

The authors from [KiC12] perform experiments to measure the effect of varying processor frequency on the energy consumption of workloads with varying levels of memory intensity on a single system. Multiple instances of each application are executed concurrently on the test system with each instance pinned to a separate core. The results of [KiC12] imply that the memory intensive tasks may be more energy efficient at slower frequencies while CPU intensive tasks may be more efficient at higher frequencies. The work in [KiC12] does not quantify memory intensity nor does it consider the effects of co-locating applications, be it instances of the same application or instances of different applications.

A study of how different architectures can affect the impact of DVFS on energy savings is analyzed in [SuH10]. Three generations of AMD processors from 2003 through 2009 are tested using a memory intensive workload across the various frequencies available to the processors. It was found that for the older processors, the most energy efficient frequencies existed in the middle of the dynamic frequency range, while for the newest processor, it was most energy efficient to run at the fastest frequency. The authors of [SuH10] only examine a single application and do not consider how performance may be affected by co-location.

The work in [TaM11] presents a study that investigates the impact of co-locating threads from multiple applications with diverse memory behavior on a quad-core system. The authors show that the execution time of co-located tasks can vary greatly depending on the other tasks that are executing. The impact co-location has on energy consumption is not considered in [TaM11], nor does the work try to isolate the effects different applications have on each other by pinning applications to specific cores.

The memory characterization of workloads from SPEC CPU2000 and SPEC CPU2006 are analyzed and presented in [Jal07]. It is shown that changes in the size of the cache of a processor can greatly affect the memory intensity of a given workload. This implies that as cache size increases, a workload can switch from being memory intensive to being CPU intensive. It is important to study the effects co-location could have on such tasks as it may make them memory intensive again once they have to contend for cache with other applications. The work in [Jal07] does not consider co-locating nor does it consider energy consumption.

A method is presented in [GoL11] to predict the performance degradation of workloads from interference due to shared processor cache (co-location). Each workload is encapsulated within its own virtual machine, and each virtual machine is pinned to its own core. The authors were able to reasonably predict the degradation due to co-location for various applications. The work in [GoL11] does not consider how the frequency of the CPU can affect the performance of the workloads nor does it examine impact of co-location on the energy consumption of individual tasks.

3. Testing Environment

3.1 Operating System

One of the goals of this research is to create a testing environment that is portable across a variety of machines and system architectures, as well as being “lightweight” to minimize noise that may occur from OS (operating system) jitter. OS jitter occurs when processes unrelated to the tested workload are executed (i.e., graphics procedures, mail daemons, security services, etc.). These processes can cause anomalies to appear in the data. The operating system used for this research is a lightweight command-line version of Ubuntu 12.04 Linux. The operating system runs Linux kernel version 3.8.0.29-generic and uses the Linux default “SCHED_OTHER” thread scheduling policy.

To ensure consistency across different experiments, any power saving features of the operating system have been disabled (e.g., screensaver and automatic processor throttling). The operating system has been configured in such a way that it can be used on a variety of different systems. Currently, it has been installed onto a bootable USB drive, but in the future, the OS could be run from a live CD or from a netboot.

3.2 Processor Performance Counters

Processors today have the ability to measure and report on numerous hardware events such as the number of cache misses or number of instructions executed through the use of performance counters. By recording different events, it is possible to gain insight into the characteristics of a given application on a given processor architecture. Typically, there are a large number of hardware events that can be measured but a small number of performance counters (e.g., in an Intel i7 there are only seven performance counters, but over 50 different measurable events [Int14]) meaning that only a limited number of events can be measured at any given time. To further complicate matters, due to differences between microarchitectures, the number, type, and availability of performance counters and measurable events can vary greatly from system to system. To allow the testing environment to be portable, we make use of additional tools to monitor and collect performance counter data across multiple systems.

The first tool used is the Performance Application Programming Interface (PAPI) which, is a portable API to hardware performance counters that simplifies interfacing between software and the native hardware performance counters of processors [Pap14]. Due to the issues involved with the variability in numbers and types of native hardware events available across microprocessor architectures, PAPI attempts to define a standard set of performance counters (“presets”) that can be found in most microprocessors. These preset performance counters try to abstract away the architecture specific details to provide an easy way to manipulate and measure similar hardware events across numerous platforms. The PAPI library contains over 100 preset performance counters that are available for use [PaE14], however as stated earlier due to differences in architectures, not all performance counters are available on every system.

The HPCtoolkit [Htk14] interfaces with PAPI and provides a set of software tools that facilitates measuring and collecting the performance data of an application. The HPCtoolkit was designed to perform benchmark testing using performance counters and therefore there is very little overhead associated with utilizing this tool. Specifically for this research, the "hpcrun-flat" tool was used to execute and monitor the test applications.

3.3 Measuring Memory Intensity

To understand the effects of co-locating multiple applications, additional metrics of performance besides execution time and energy consumption should be measured. One hypothesis of this research is that applications that frequently need to access the last-level of cache and main memory have a greater degradation in performance when co-located with other memory intensive applications compared to applications that are mostly CPU intensive. To test this hypothesis, the measure we use here for relative memory intensity is last-level cache misses divided by instructions executed.

Relative memory intensity indicates how often an application must access main memory (DRAM) per instruction executed. The number of times an application accesses main memory can be measured by keeping track of the number of last-level cache misses. The total number of last-level cache misses is, however, not enough to classify the memory intensity of an application, as different applications can execute for different amounts of time, and have different instruction counts. Therefore, we normalize our last-level cache misses by the number of total instructions executed. The resulting metric called relative memory intensity allows for the comparison of memory intensity across applications regardless of the instruction counts of the applications.

The number of last-level cache misses and the total number of instructions executed are measured using performance counters. PAPI does not contain a standard "last-level cache miss" performance counter and thus the appropriate level (L2 or L3) cache miss event must be set depending on the specific microprocessor architecture.

3.4 Processor Performance States (P-states)

Performance states (P-states) utilize dynamic voltage and frequency (DVFS) capability in processors to control the power consumption and speed at which the processor is operating. P-states are denoted by integer numbers, with P-state P0 indicating the highest voltage and highest (fastest) frequency. Higher P-state numbers indicate lower voltages and lower (slower) CPU frequencies. The number of P-states available for any given CPU, as well as the voltage and frequency pairings that each P-state represents, are different for every processor architecture, and therefore trends and behaviors of applications on one system may be significantly different when compared to another system. Generally, across different processor microarchitectures, operating in higher P-states results in increased execution time.

Furthermore, P-states only control the portion of a processors total power consumption called the dynamic power. The remaining power in the processor is called the static power and is assumed to remain constant regardless of the P-state. The ratio of dynamic power to static power differs from architecture to architecture, and using P-states to decrease the CPU frequency and voltage will decrease the dynamic power use but also likely increases an application's execution time. Due to static power being constant across P-states and increased execution time, there is no guarantee that operating in a slower P-state will result in reduced energy consumption.

This research provides an infrastructure for analyzing how the performance in terms of execution time, energy consumption, and memory intensity changes across different P-states, in modern multi-core processor architectures.

Table 1: PARSEC Applications

Application	Description
canneal	cache-aware simulated annealing to optimize routing cost of a chip design
streamcluster	online clustering of an input stream
blackscholes	option pricing with Black-Scholes Partial Differential Equation (PDE)
bodytrack	body tracking of a person

3.5 "Watts Up? PRO" Power Meter

We used a *Watts Up? PRO* power meter [Wat14] to collect and calculate the energy used while an application executes on microprocessors. The *Watts Up?* meter connects to and measures the system at the "outlet" level, meaning that all the power used by the whole system is measured. The meter is able to measure the instantaneous power draw of the system at one second intervals. To calculate the energy consumed by an application, the power samples, which are the average power over the one-second intervals and so are simply summed over the execution time.

4. Experimental Setup

4.1 PARSEC Benchmark Suite

The workloads used in this research are taken from the PARSEC benchmarking suite [Par14], which consists of a diverse set of applications from many different computing areas. Four benchmarking applications were chosen from the PARSEC suite with the intention of providing a representative set of applications having varying degrees of memory intensity. The applications chosen for experimentation, along with a brief description, can be found in Table 1, organized from most memory intensive to least memory intensive.

4.2 Co-location Experiments

To measure the effect of co-location on the energy consumption, execution time, and relative memory intensity of an application, a set of baseline tests were conducted to establish measurements that these three metrics could be compared to, across changes in the number of processor cores. For a baseline test, the applications were executed by themselves on the multi-core processor. During execution, the application was pinned to a specific core using the Linux "taskset" command and was the only process executing other than OS-related processes. Additionally, each baseline test was executed at different processor frequencies (P-states) ranging from 3.40GHz, the CPUs default highest speed, down to 1.70GHz. Energy consumption, execution time, and relative memory intensity were measured for each application in each P-state.

From the relative memory intensity results of the baseline tests (Table 2), the applications were then classified into three groups based on the magnitude of their relative memory intensities. The applications are shown in order of decreasing memory intensity. As indicated in the table, canneal is the most memory intensive application with over two last-level cache misses per 100 instructions executed while blackscholes is the least memory intensive application with fewer than one last-level cache miss per 100,000 instructions executed.

Table 2: Memory Intensity Classification

Applications	Classification	Relative Memory Intensity
canneal	Intensity III	2.25×10^{-2}
streamcluster	Intensity III	1.64×10^{-2}
blackscholes	Intensity II	2.29×10^{-5}
bodytrack	Intensity I	7.44×10^{-6}

The canneal and streamcluster applications exhibit the most interactions with main memory, and are categorized as being "Intensity III" memory intensive tests, blackscholes is categorized a "Intensity II" memory intensive application, and the bodytrack application, accessing memory the least, is categorized as being a "Intensity I" memory intensive application.

To test the effect of co-location on energy use, execution time, and relative memory intensity, two additional sets of experiments were performed. The first set contains experiments where two applications were co-located each application is pinned to its own core. Each application could be paired with one of the other applications or a copy of itself. For brevity, a subset of these possible pairs is presented in Table 3. The selection of this subset of pairs is due to the fact the canneal application is the most memory intensive application that was tested. canneal is therefore tested against the other applications to determine how those applications are impacted when co-running with a highly memory intensive application.

The second set of experiments increases the level of co-location from two applications up to four applications, using the "taskset" command to pin each application to its own core within the quad-core processor. Again, a subset of the numerous possible combinations for co-locating the applications is presented in this work for brevity. These subset combinations are shown in Table 4.

Table 3: Two Core Interference Tests

Test Type	Applications co-located together
2 Intensity III	canneal, streamcluster
1 Intensity III and 1 Intensity II	canneal, blackscholes
1 Intensity III and 1 Intensity I	canneal, bodytrack

Table 4: Four Core Interference Tests

Test Type	Applications co-located together
4 Intensity III	2 canneal, 2 streamcluster
1 Intensity III, 1 Intensity II, and 2 Intensity I	1 canneal, 1 blackscholes, 2 bodytrack
2 Intensity III and 2 Intensity I	1 canneal, 1 streamcluster, 2 bodytrack

5. Results

For the results presented in this section, tests were performed on the 64-bit Intel i7 3770 3.40 GHz quad-core processor [Int12]. In this processor, each core has its own private L1 and L2 caches, but a shared (8MB) L3 cache, that is shared by all four cores. The operating frequency of the processor can vary from 3.40 GHz to 1.70 GHz over the range of P-states, and because DVFS is used to do this, it prevents the processor from over-clocking the CPU into Intel's "turbo" mode during the baseline tests when only one core is being used. The average of nine runs of each test are reported.

The results of the co-location tests are shown in Figures 1 to 3. Each figure contains the results for relative memory intensity, execution time, and energy consumption of a single application across multiple processor frequencies for the baseline (red), two-way co-location (green), and four-way co-location (blue) tests. Subfigure (a) of each figure shows the relative memory intensity, Subfigure (b) shows the execution time, and Subfigure (c) shows the energy consumption.

Note that for the memory intensity and execution time subfigures (Subfigures 1(a) and (b) through Subfigures 4(a) and (b), the results shown are for the individual application being presented. For example, when looking at the execution times of canneal in Subfigure 1(b), the execution times for "2 Intensity III" and "4 Intensity III" bars are the actual execution times of only the canneal application, not the execution times of all the co-located applications. In contrast, due to the fact the *Watts up? PRO* meter takes measurements at the "outlet" level, the energy results in Subfigures 1(c) through 4(c) show the energy consumption for all co-located applications. Furthermore, because the applications have different execution times, the energy consumed that is reported for an application is the total energy consumed during that application's execution time.

Figures 1 and 2 contain the results for the canneal and streamcluster applications. Recall that canneal and streamcluster are the two "Intensity III" memory intensive applications that were tested. When examining the effect of co-location on the relative memory intensity Subfigures 1(a) and 2(a) it is apparent that when multiple Intensity III applications are co-located the number of last-level cache misses increase, resulting in more memory accesses, and implying that the applications are creating interference by evicting lines from the last-level cache that were being used by other applications. The large variations between the baseline results and the dual and quad-core interference results seen between every application is a result of differences in how memory interference affects each application. It should be noted that the scale of each graph changes between figures. The execution time of both applications are shown in Subfigure 1(b) and Subfigure 2(b) respectively. Co-location increases the execution time of applications.

The results for the blackscholes and bodytrack applications are shown in Figures 3 and 4, respectively. These two application are significantly less memory intensive than canneal and streamcluster. The results for relative memory intensity (Subfigures 3(a) and 4(a)) indicate that co-location can greatly increase the memory intensity of "Intensity I" memory intensive tasks in a relative sense, but it is important to note that even in the worst case (highest memory intensity) these two applications still have fewer last-level cache misses than either of the best cases (lowest memory intensity) for the canneal and streamcluster applications. This is also the reason

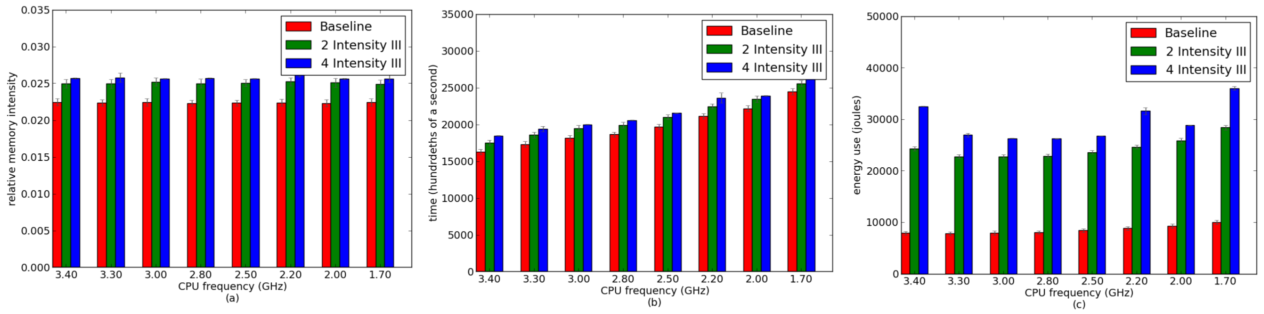

Fig. 1: The baseline (red), two-way co-location (green), and four-way co-location(blue) tests for canneal. The x-axis of each subfigure is the frequency of the processor. (a) Memory intensity results, the y-axis is the relative memory intensity measured as the number of last-level cache misses per instruction executed. (b) Exectuion time results, the y-axis is the execution time measured in hundredths of a second. (c) Energy consumption results, the y-axis is the energy consumed measured in Joules.

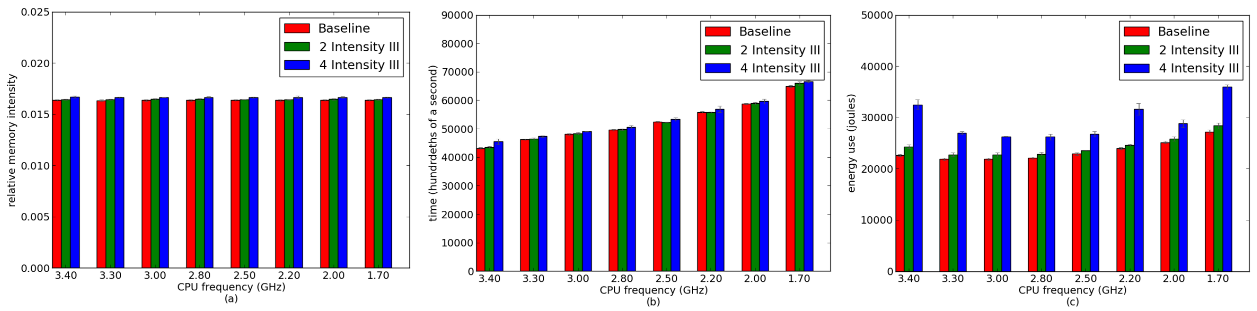

Fig. 2: The baseline (red), two-way co-location (green), and four-way co-location(blue) tests for streamcluster. The x-axis of each subfigure is the frequency of the processor. (a) Memory intensity results, the y-axis is the relative memory intensity measured as the number of last-level cache misses per instruction executed. (b) Exectuion time results, the y-axis is the execution time measured in hundredths of a second. (c) Energy consumption results, the y-axis is the energy consumed measured in Joules.

for the apparent fluctuations in the memory intensity values for blackscholes and bodytrack. As can be seen by examining the variations shown in the error bars, even the smallest variations present in the highly memory intensive canneal data is still greater than the largest variations in the blackscholes data (a difference of 7.05×10^{-5} for canneal as compared to 6.16×10^{-5} for blackscholes). Further analysis of the execution times for blackscholes and bodytrack (Subfigures 3(b) and 4(b)) indicates that even though co-location may have a significant relative increase in memory intensity for these applications, it minimally affects the execution times.

In general, across all applications, changing processor frequency (P-state) has minimal impact on the relative memory intensity (Subfigures 1(a), 2(a), 3(a), 4(a)), as processor speed should not affect the memory behavior of an application. Changing processor frequency does have a noticeable impact on execution time (Subfigures 1(b), 2(b), 3(b), 4(b)). As expected, when the frequency decreased, the execution times of all the applications increased. This increase in execution time is not uniform across all the applications. In general, the "Intensity I" memory intensive tasks are more sensitive to changes in frequency, meaning they experience a larger percent increase in execution time compared to the "Intensity III" memory intensive tasks. Most likely this is due to the fact that the "Intensity I" memory intensive tasks perform most of their computing on the CPU, operating mainly out of the cache, thus they do not need to access main memory very often.

An important distinction to make when analyzing the energy results (Subfigures 1(c), 2(c), 3(c), 4(c)) of these experiments is that the baseline results show the energy consumed during a single task's execution, while the co-location results show the energy consumed for *multiple* task's execution. For example in Figure 1(c) the energy use shown in the graph is a measure of the entire test's energy use up until the time that the canneal application finishes executing, whereas while the energy use of streamcluster in Figure 2(c) is taken from the same data measurements as that of Figure 1(c) the execution time of the the streamcluster application is longer, so its power use is greater than that of canneal.

It can be observed that even though the energy consumed to execute a task increases when co-located with other tasks, the total energy used to completely execute all the tasks actually decreases. For example, as detailed in table 5, if canneal and streamcluster were executed independently without any co-location they would require approximately 30,509 Joules total to run, but when co-located with one another they require approximately only 24,281 Joules total. This is because when run independently, both applications will separately incur the static energy present in the system in addition to the dynamic energy used during their execution. When co-located, both applications are able to share the

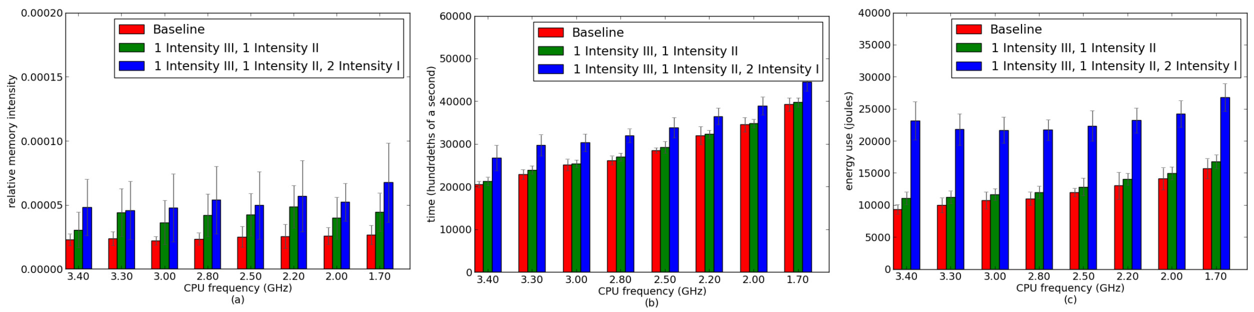

Fig. 3: The baseline (red), two-way co-location (green), and four-way co-location(blue) tests for blackscholes. The x-axis of each subfigure is the frequency of the processor. (a) Memory intensity results, the y-axis is the relative memory intensity measured as the number of last-level cache misses per instruction executed. (b) Execution time results, the y-axis is the execution time measured in hundredths of a second. (c) Energy consumption results, the y-axis is the energy consumed measured in Joules.

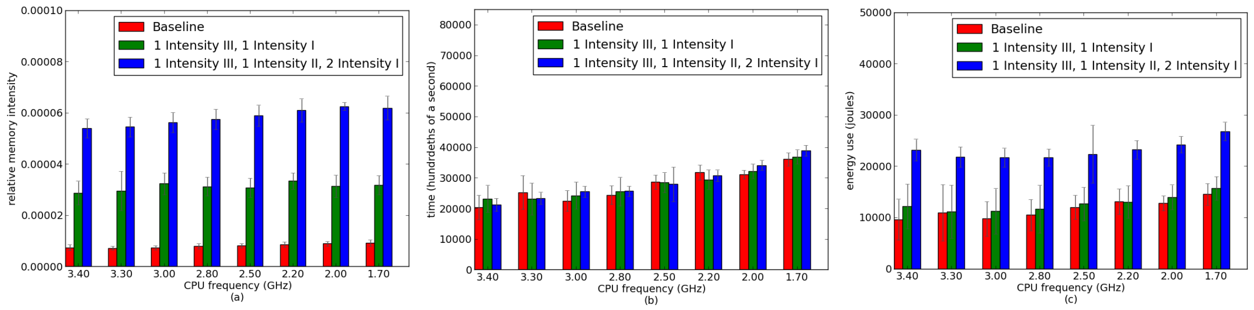

Fig. 4: The baseline (red), two-way co-location (green), and four-way co-location(blue) tests for bodytrack. The x-axis of each subfigure is the frequency of the processor. (a) Memory intensity results, the y-axis is the relative memory intensity measured as the number of last-level cache misses per instruction executed. (b) Exectuion time results, the y-axis is the execution time measured in hundredths of a second. (c) Energy consumption results, the y-axis is the energy consumed measured in Joules.

static power resulting in less static energy being consumed during execution, and less total energy used overall. Only a subset of these results are shown in the table, but they are consistent across all test runs. Furthermore, from the Subfigures 1(c), 2(c), 3(c), 4(c) it can be seen that the optimal p-state for minimizing dynamic energy is different between applications, and can change between an application's baseline, two-core, and four-core co-location tests.

This behavior along with the fact that minimal performance degradation occurs when "Intensity III" and "Intensity I" memory intensive tasks are co-located with one another provides many interesting and exciting possibilities for smart resource allocation managers in high performance computing (HPC) systems. For example, task schedulers could use this information to intelligently co-locate applications with differing memory intensities on multi-core processors in server nodes to minimize performance loss and decrease system energy consumption.

Table 5: Test Energy Savings from Sharing Static Power (units in Joules, results are taken from tests run at 3.40GHz)

Application	Energy Use
canneal	7873
streamcluster	22,636
blackscholes	9347
bodytrack	9583
canneal + blackscholes	11,055 (Co-located)
canneal + streamcluster	24,281 (Co-located)
canneal + blackscholes + 2 bodytrack	28,729 (Co-located)
2 canneal + 2 streamcluster	32,504 (Co-located)

6. Conclusion and Future Work

With the desire for increased performance in computing systems, multi-core processors have become a popular and prevalent method of achieving higher performance. These multi-core processors are able to execute multiple applications at one time, however there exist complex interactions between the memory access behavior of applications that may cause degradations in performance and increased energy consumption for each individual application. This work examined the impact of memory interference on execution time, energy consumption, and relative memory intensity for co-located applications on multi-core processors. Specifically, a portable and lightweight testing framework was presented where four workloads taken from the PARSEC benchmark suite were run on an Intel i7 quad-core machine. The results verify that applications that have "Intensity III" memory intensity are more sensitive in terms of execution time and energy consumption when co-located with other "Intensity III" memory intensive applications, while "Intensity I" memory intensive applications are much less susceptible to degradations in their performance when co-located with other applications. For the specific system tested, it was shown that the optimal processor frequency for minimizing energy consumption could change based on the application and the co-location workload due to variations in use of dynamic power, the increased execution time, and static power present within the system. It was also found that co-running applications can also lead to sharing of static power use among the simultaneously running applications, which ends up decreasing the energy consumed for each application. Changing the frequency of the processor had negligible effect on the memory intensity of the applications.

Some possible directions for future work include increasing the number of PARSEC test applications as well as including applications from additional benchmark suites and performing co-location tests on a variety of systems. Information gathered from these tests could then be used by high performance scheduling systems to co-locate applications in a manner that minimizes performance degradation and energy consumption, as well as being extended to resource management in heterogeneous multicore-based distributed systems; e.g., [ApY11], [YoA13], [YoP13].

7. Acknowledgements

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, supported by the Extreme Scale Systems Center at ORNL, which is supported by the Department of Defense. This work was supported by the National Science Foundation (NSF) under grant numbers CNS-0905339, CCF-1252500, CCF-1302693, and an NSF Graduate Research Fellowship. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. The authors thank M. Hilton, B. Khemka, S. Powers, R. Rambharos, and M. Wright for their valuable comments on this work.

References

[ApY11] J. Apodaca, D. Young, L. Briceño, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, S. Bahirat, B. Khemka, A. Ramirez, and Y. Zou, "Stochastically robust static resource allocation for energy minimization with a makespan constraint in a heterogeneous computing environment," *9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '11)*, Dec. 2011, pp. 22–31.

[DcD12] (2012) 2012 DatacenterDynamics Industry Census, http://www.datacenterdynamics.com/blogs/industry-census-2012-emerging-data-center-markets.

[FrB13] R. Friese, T. Brinks, C. Oliver, A. A. Maciejewski, H. J. Siegel, and S. Pasricha, "A machine-by-machine analysis of a bi-objective resource allocation problems," *The 2013 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2013)*, July 2013, pp. 3–9.

[FrK13] R. Friese, B. Khemka, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, and S. W. Poole, "An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environments," *22nd Heterogeneity in Computing Workshop (HCW 2013), in the proceedings of the IPDPS 2013 Workshops & PhD Forum (IPDPSW)*, May 2013.

[GoL11] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines," *2nd Symposium on Cloud Computing (SOCC'11)*, 2011, pp. 1–14.

[Htk14] (accessed Mar. 2014) HPCToolkit, http://hpctoolkit.org/.

[Int12] Intel. (2012) Intel core i7-3770 processor, http://ark.intel.com/products/65719/.

[Int14] "Intel 64 and ia-32 architectures software developer's manual volume 3b: System programming guide, part 2," Tech. Rep., Feb 2014, http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf.

[Jal07] A. Jaleel, "Memory characterization of workloads using intrumentation-driven simulation," Tech. Rep., 2007, http://www.jaleels.org/ajaleel/workload/SPECanalysis.pdf.

[KiC12] S. Kim, C. Choi, H. Eom, and H. Y. Yeom, "Energy-centric DVFS controlling method for multi-core platforms," *5th International Workshop on Multi-Core Computing Systems (MuCoCoS'12), as part of Super Computing 2012*, Nov 2012.

[OxP13] M. Oxley, S. Pasricha, H. J. Siegel, and A. A. Maciejewski, "Energy and deadline constrained robust stochastic static resource alloation," *1st Workshop on Power and Energy Aspects of Computation (PEAC 2013), in the proceedings of the 10th International Conference on Parallel Processing and Applied Mathematics (PPAM 2013)*, Sep 2013, p. 10.

[PaE14] (accessed Mar. 2014) PAPI events by architectures, http://icl.cs.utk.edu/projects/papi/presets.html.

[Pap14] (accessed Mar. 2014) Performance application programming interface, http://icl.cs.utk.edu/papi/.

[Par14] (accessed Mar. 2014) PARSEC benchmark suite, http://parsec.cs.princeton.edu/.

[SuH10] E. L. Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," *The 2010 International Conference on Power Aware Computing and Systems (HotPower '10)*, Oct 2010, p. 5.

[TaM11] L. Tang, J. Mars, N. Vachharajani, R. Hundt, and M. L. Soffa, "The impact of memory subsystem resource sharing on datacenter applications," *38th Annual International Symposium on Computer Architecture (ISCA'11)*, June 2011, pp. 283–294.

[Wat14] (accessed Mar. 2014) Watts Up? plug load meters, https://www.wattsupmeters.com/secure/products.php?pn=0.

[YoA13] B. D. Young, J. Apodaca, L. D. Briceño, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou, "Deadline and energy constrained dynamic resource allocation in a heterogeneous computing environments," *The Journal of Supercomputing*, Vol. 63, No. 2, Feb. 2013, pp. 326–347.

[YoP13] D. Young, S. Pasricha, A. A. Maciejewski, H. J. Siegel, and J. T. Smith, "Heterogeneous energy and makespan-constrained dag scheduling," *International Workshop on Energy Efficient High Performance Parallel and Distributed Computing (EEHPDC-2013), sponsor: ACM*, Jun. 2013, pp. 3–11.